Inventory Operations notebooks

Script generates the inventory operations data to analyse inventory movements and restock logistics

Inventory Distribution and Management Data

1. This is a generated data sample that relates to realistic inventory management and distribution processes
2. For stock levels, restocking behavior, lead times, and geographic consistency
3. Further modelling, transformations, and calculated columns, and measures are implemented downstream(Dataflows, Semantic Model - DAX)

In [ ]:

```
import pandas as pd
import numpy as np
import random
from datetime import datetime, timedelta

from notebookutils import mssparkutils  # Only works in Microsoft Fabric

%pip install faker

from faker import Faker

# --- Configuration ---
n = 5000
np.random.seed(42)
random.seed(42)

# Territories with lat/lon
territories = {
    "North": [
        ("Manchester", 53.4808, -2.2426),
        ("Leeds", 53.8008, -1.5491),
        ("Newcastle", 54.9784, -1.6174),
        ("Liverpool", 53.4084, -2.9916)
    ],
    "South": [
        ("London", 51.5074, -0.1278),
        ("Brighton", 50.8225, -0.1372),
        ("Oxford", 51.7520, -1.2577),
        ("Reading", 51.4543, -0.9781)
    ],
    "East": [
        ("Cambridge", 52.2053, 0.1218),
        ("Norwich", 52.6309, 1.2974),
        ("Ipswich", 52.0567, 1.1482)
    ],
    "West": [
        ("Bristol", 51.4545, -2.5879),
        ("Exeter", 50.7184, -3.5339),
        ("Bath", 51.3758, -2.3599),
```

```python
        ("Cardiff", 51.4816, -3.1791)
    ]
}

warehouses = {"North": "WH1", "South": "WH2", "East": "WH3", "West": "WH4"}
suppliers = ["Wickes", "Screwfix", "Toolstation"]

products = {
    "Electronics": ["Widget A", "Widget B", "Gizmo C"],
    "Home": ["Gadget X", "Gadget Y", "Gadget Z"],
    "Hardware": ["Tool Z", "Tool Y", "Kit A"],
    "Office": ["Desk Lamp", "Chair Pro", "Monitor Stand"]
}

# --- Helper functions ---
def random_date(start, end):
    delta = end - start
    return start + timedelta(days=np.random.randint(0, delta.days))

def choose_territory(region):
    return random.choice(territories[region])

def choose_product():
    category = np.random.choice(list(products.keys()), p=[0.3, 0.3, 0.25, 0.15])
    product = random.choice(products[category])
    return category, product

# --- Generate data ---
rows = []
start_date = datetime(2023, 1, 1)
end_date = datetime(2025, 10, 1)

for i in range(1, n+1):
    pid = f"P{i:04d}"
    region = np.random.choice(list(territories.keys()))
    territory, lat, lon = choose_territory(region)
    warehouse = warehouses[region]
    supplier = random.choice(suppliers)
    lead_time = int(np.random.normal(loc={"Wickes":5,"Screwfix":6,"Toolstation":
    lead_time = max(2, lead_time)
    category, product_name = choose_product()
    # Stock distribution varies by category
    stock_level = {
        "Electronics": np.random.randint(50, 200),
        "Home": np.random.randint(80, 250),
        "Hardware": np.random.randint(100, 400),
        "Office": np.random.randint(60, 300)
    }[category]
    reorder_point = int(stock_level * np.random.uniform(0.4, 0.6))
    last_restock = random_date(start_date, end_date)
    needs_restock = "Yes" if stock_level < reorder_point else "No"
    row = {
        "ProductID": pid,
        "ProductName": product_name,
        "Category": category,
        "StockLevel": stock_level,
        "ReorderPoint": reorder_point,
        "LastRestockDate": last_restock,
        "Warehouse": warehouse,
        "Region": region,
        "Territory": territory,
        "Supplier": supplier,
        "LeadTimeDays": lead_time,
        "Latitude": lat,
        "Longitude": lon,
        "Country": "United Kingdom",
        "Continent": "Europe",
        "NeedsRestock": needs_restock
    }
    rows.append(row)

df = pd.DataFrame(rows)
df.to_csv("inventory_operations_data.csv", index=False)

#convert to spark dataframe
df_spark = spark.createDataFrame(df)

df_spark.write.mode("overwrite").saveAsTable("inventory_operations_data")

#I want the file saved in the base 'Files' folder as is - needs a few steps urgg
df_spark.coalesce(1) \
    .write.mode("overwrite") \
    .option("header", True) \
    .csv("Files/tmp_inventory_operations_data")

# List files in the temp folder
files = mssparkutils.fs.ls("Files/tmp_inventory_operations_data")    # only a mi

# Find the part file (the real CSV)
csv_file_path = [f.path for f in files if f.path.endswith(".csv")][0]


# Source and destination paths
source_path = "Files/tmp_inventory_operations_data"
dest_path = "Files/inventory_operations_data.csv"

# Delete existing destination if it exists
if mssparkutils.fs.exists(dest_path):
    print(f"Deleting existing file at {dest_path} ...")
    mssparkutils.fs.rm(dest_path, recurse=True)

# Move file from temp folder to destination
print(f"Moving {source_path} -> {dest_path} ...")

# Move and rename to customer_data.csv in the root Files folder
mssparkutils.fs.mv(csv_file_path, dest_path)

# Optional: clean up temp folder
mssparkutils.fs.rm(source_path, recurse=True)


print("✅ Generated 'inventory_operations_data.csv' with 5000 rows and NeedsRest
```

Query to preview our created SQL table

In [2]:
```
df = spark.sql("SELECT * FROM laserengravelakehouse.inventory_operations_data LI
display(df)
```

```
StatementMeta(, 66d3be0c-0f2c-4aa9-91e3-9f5e38f0672b, 10, Finished, Available, Fi
nished)
```

```
SynapseWidget(Synapse.DataFrame, 7a75fd35-cd0f-4810-8328-a870bbdd68db)
```